



# National Center for Genome Analysis Support

## Indiana University Pervasive Technology Institute

Carrie Ganote ◀  
Sheri Sanders ◀  
Tom Doak ◀  
◀ Indiana University

Cicada Brokaw ◀  
Bhavya N P ◀  
Brian Haas ◀  
◀ Broad Institute

Asma Bankapur ◀  
Tim Tickle ◀  
Phil Blood ◀  
◀ Pittsburgh Supercomputing Center

## Pushing the limits of job flexibility on HPC

### Mounted file systems

NCGAS utilized IU's wide-area network Lustre file system - the Data Capacitor (WAN) - to share files between IU and Pittsburgh Supercomputing Center (PSC). It used UID mapping and permissions sets to translate the ownership of files for users across systems without requiring the same username and uid to exist on both. This effort was retired along with Blacklight.

New effort is being put forth for sharing jobs as well as files to allow large assembly jobs to run on partner PSC's new large-memory machine, Bridges. We are still in the beginning phase of testing these connections, but it will resolve several challenges:

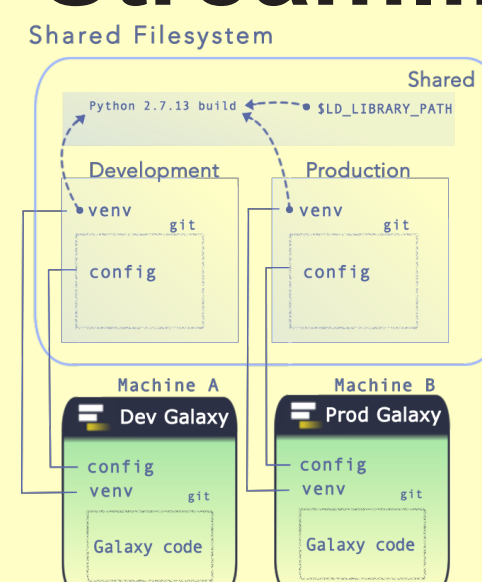
1. File movement between the mount points will be much faster, easier and automated. The job doesn't have to wait on the entire file to transfer before starting.
2. Many of our users would like to be able to assemble more data, but don't want to have to move data, create new accounts, and learn new systems.

### RAMdisk Improvements

Trinity can be tough on file systems - especially when the file system is over a network LAN or sshfs. In such cases, it can be helpful to minimize the amount of read/write done over the network. Carefully choosing the working directory can be important for tool performance. This can be done by utilizing local disk or in our case, running on RAMdisk.

RAMdisk is a virtual file system that exists entirely in memory. Our cluster allots half of the memory to file I/O. Trinity allows for alternative locations for job running in its configuration, but it would be extremely helpful to have as a Galaxy configuration on the job runner level.

### Streamlining Development



Our approach to keeping Galaxy updated and happy is to keep the Development Galaxy and the Production Galaxy as separate as possible while still maintaining equivalence. Config and python environment are pulled out of Galaxy git control and put into our own git repo, which we can then push and pull between from dev to prod.

### Job Checkpoints

Many software tools benefit from checkpointing, or starting from the last point they left off. This is especially important for resource-hungry, long-running jobs with multiple steps. It can be extremely helpful too when partitioning a job into multiple pieces for running in different hardware settings.

We do this for Trinity by allowing the user to create a tag for their job. This tag, along with their username, is used as a directory name in a new working directory. Input files and the working directory are changed inside the wrapper script, and final files are placed in the location Galaxy expects them to be in at the conclusion of the job.

The next step is to make this automatic by generating an id for the job, creating a new working directory using that id, and storing it as a hidden parameter for the job. It will be recovered when the job is rerun through the galaxy interface, without the user needing to know to tag the job beforehand.

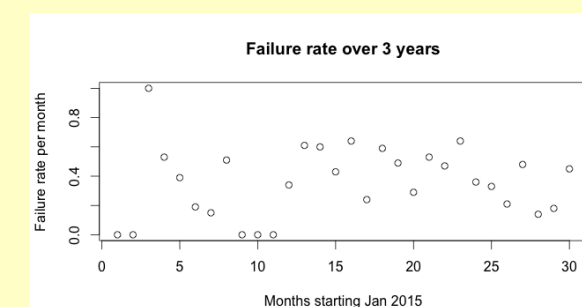
### Job Retry

Jobs can fail for reasons which can't be reproduced - events such as network latency and file system evictions can lead to job failures that could be prevented if it were tried again. This is especially helpful for checkpointing programs that can detect where the job left off. Since a retry takes place within the same job, no further need for copying files to new locations is needed. We give Trinity 2 tries before calling quits.

To implement this, a wrapper script around the current Trinity tool parses the arguments and forks a subprocess to run Trinity. The script examines the exit code of the Trinity job, as well as the output of the log file, to determine whether the job was a success. If not, the retry count decrements until it is below a threshold and the job is allowed to fail.

### Results

It is early to tell if the improvements made to the Trinity tool have been effective in alleviating all job issues. There are still cases where, even after rerunning with a checkpoint, a tool can fail repeatedly on a particular input. We may be seeing a downward trend in error rate, but there is more work needed for a perfectly robust system.



### Dynamic Job Runner & Job Limits

Trinity CTAT Galaxy takes advantage of many of the built-in features available for fine-tuning the deployment of jobs on our systems. One very useful feature is dynamic job running; we use this primarily to allocate resources appropriate to the size of the file inputs for the job. We can then tie the job runner destinations to tags that help us use job limits to avoid overwhelming the system with huge jobs. Using the tool destinations vaml file will be the next step for us.

This material is based upon work supported by the National Science Foundation under Grant Nos. DBI-1458641, DBI-1458689, CNS-0521433.  
- Goecks, J, Nekrutenko, A, Taylor, J and The Galaxy Team. "Galaxy: a comprehensive approach for supporting accessible, reproducible, and transparent computational research in the life sciences." Genome Biol. 2010 Aug 25;11(8):R86.  
- Blankenberg D, Von Kuster G, Coraor N, Ananda G, Lazarus R, Mangan M, Nekrutenko A, Taylor J. "Galaxy: a web-based genome analysis tool for experimentalists". Current Protocols in Molecular Biology. 2010 Jan; Chapter 19:Unit 19.10.1-21.  
- Giardine B, Riemer C, Hardison RC, Burhans R, Elmtski L, Shah P, Zhang Y, Blankenberg D, Albert I, Taylor J, Miller W, Kent WJ, Nekrutenko A. "Galaxy: a platform for interactive large-scale genome analysis." Genome Research. 2005 Oct; 15(10):1451-5.